Tópico 03 - Sistemas operacionais e compiladores

Hugo Vinícius Leão e Silva

hugovlsilva@gmail.com, hugo.vinicius.16@gmail.com, hugovinicius@ifg.edu.br

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Campus Anápolis
Curso de Bacharelado em Ciência da Computação

19 de junho de 2023

Visão geral

Preliminares

- Computador é formado por *hardware* e *software*;
- Hardware é a parte *física* do computador;
- Software é a parte *lógica* do computador, são os programas de computador. São basicamente divididos em:
  - Programas aplicativos;
  - Sistemas operacionais.
- Firmware é um tipo específico de *software* que se situa diretamente no *hardware*. Normalmente é gravado num chip de memória ROM (do inglês *Read-Only Memory*).

- Um sistema operacional (SO) pode ser visto como:
  - Um gerente de recursos de sistema;
  - Uma interface de usuário (UI, do inglês *User Interface*);
  - Um conjunto de programas que suportam (no sentido de auxiliar) a execução dos programas de aplicativo.

Sistemas Operacionais

- Um sistema operacional (SO) pode ser visto como:
  - (FLYNN e MCHOES, 2002):
    - "**Gerente executivo (...) que administra todos os componentes de hardware e software**";
    - "(...) controla cada arquivo, dispositivo [de E/S], seção de memória RAM e nanossegundo de tempo de processamento. Controla quem pode utilizar o sistema e de que maneira".
  - (TANEMBAUM, 2010):
    - "Sistema computacional (...) consiste em um ou mais processadores, memória principal, discos, impressoras, (...). **Se cada programador de aplicações tivesse de entender como tudo isso funciona em detalhes, nenhum código chegaria a ser escrito**";
    - "(...) sistema operacional, cujo trabalho é fornecer aos programas de usuário um **modelo de computador** melhor, mais simples e mais limpo (...)".

Tipos/evolução dos Sistemas Operacionais

- **Sistemas em lote**:
  - Década de 1950;
  - Usados em *mainframes*;
  - Cada *job* (ou tarefa) era enviado pelo programador para o operador na sala de operações junto com os cartões perfurados;
  - A função desses sistemas era apenas garantir que os recursos computacionais fossem transferidos de um *job* para outro.
- **Sistemas de compartilhamento de tempo**:
  - Surgiu o conceito de multiprogramação – vários processos na RAM;
  - Com a multiprogramação veio o conceito de compartilhamento de tempo dos recursos computacionais;
  - Somente alocava os recursos necessários para cada *job* desde que estivessem disponíveis;
  - O usuário interage diretamente com o SO;

Tipos/evolução dos Sistemas Operacionais

- **Sistemas paralelos**:
  - São sistemas capazes de alocar eficientemente várias CPUs para os processos, aumentando a velocidade de processamento.
- **Sistemas distribuídos**:
  - Com o surgimento das redes de computadores, surgiram esse tipo de SO, que é capaz de gerenciar recursos computacionais de diversos computadores ligados em rede e com controle de segurança.
- **Sistemas de tempo real**:
  - São sistemas que realiza tarefas dentro em sistemas com restrições de tempo de resposta.
- **Sistemas pessoais**:
  - São os sistemas utilizados nos dispositivos pessoais como PCs, tablets, smartphones etc.

Carregamento de um SO

- Uma das funções do SO é carregar programas aplicativos em memória RAM;
- Um SO também é um software. Ao ligar o computador, quem carrega o SO na RAM?

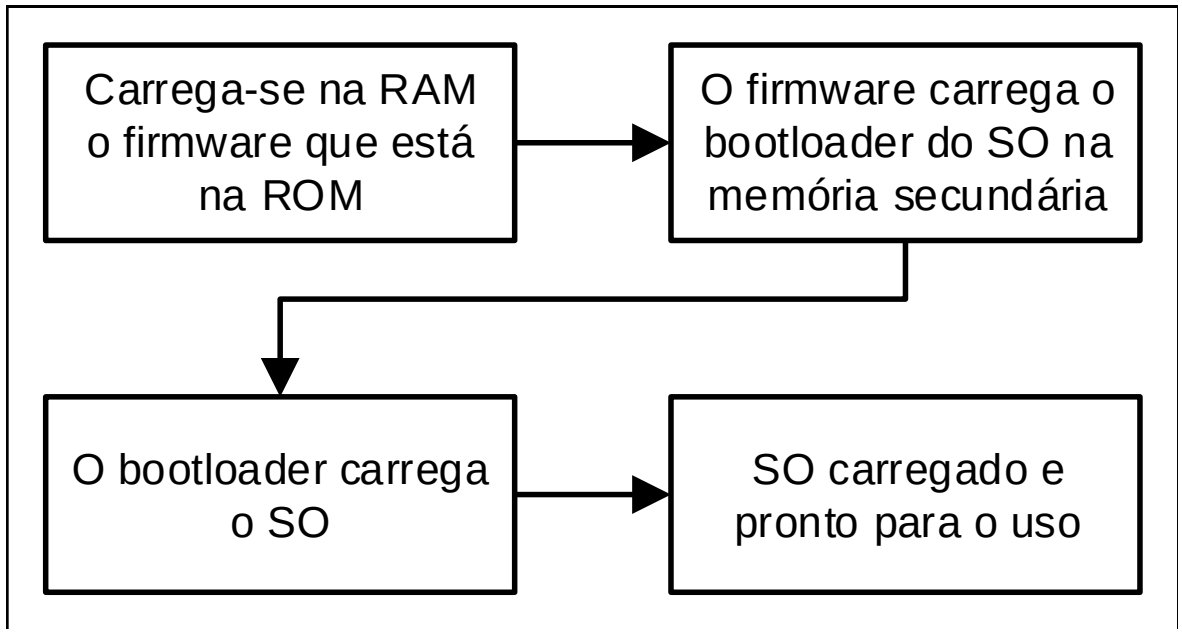


Figura: Processo de carregamento de um SO

Componentes de um SO - *overview*

- Cada gerenciador é responsável por um conjunto de tarefas;
- Mas eles não agem sozinhos e às vezes têm que agir de maneira coordenada;
- Basicamente são quatro gerenciadores básicos:
  - Gerenciador de memória;
  - Gerenciador de processos;
  - Gerenciador de dispositivos de E/S;
  - Gerenciador de arquivos;
- SOs que suportam redes de computadores também têm o gerenciador de redes;
- E, claro, a interface de usuário é um componente essencial de um SO.

Componentes de um SO - *overview*

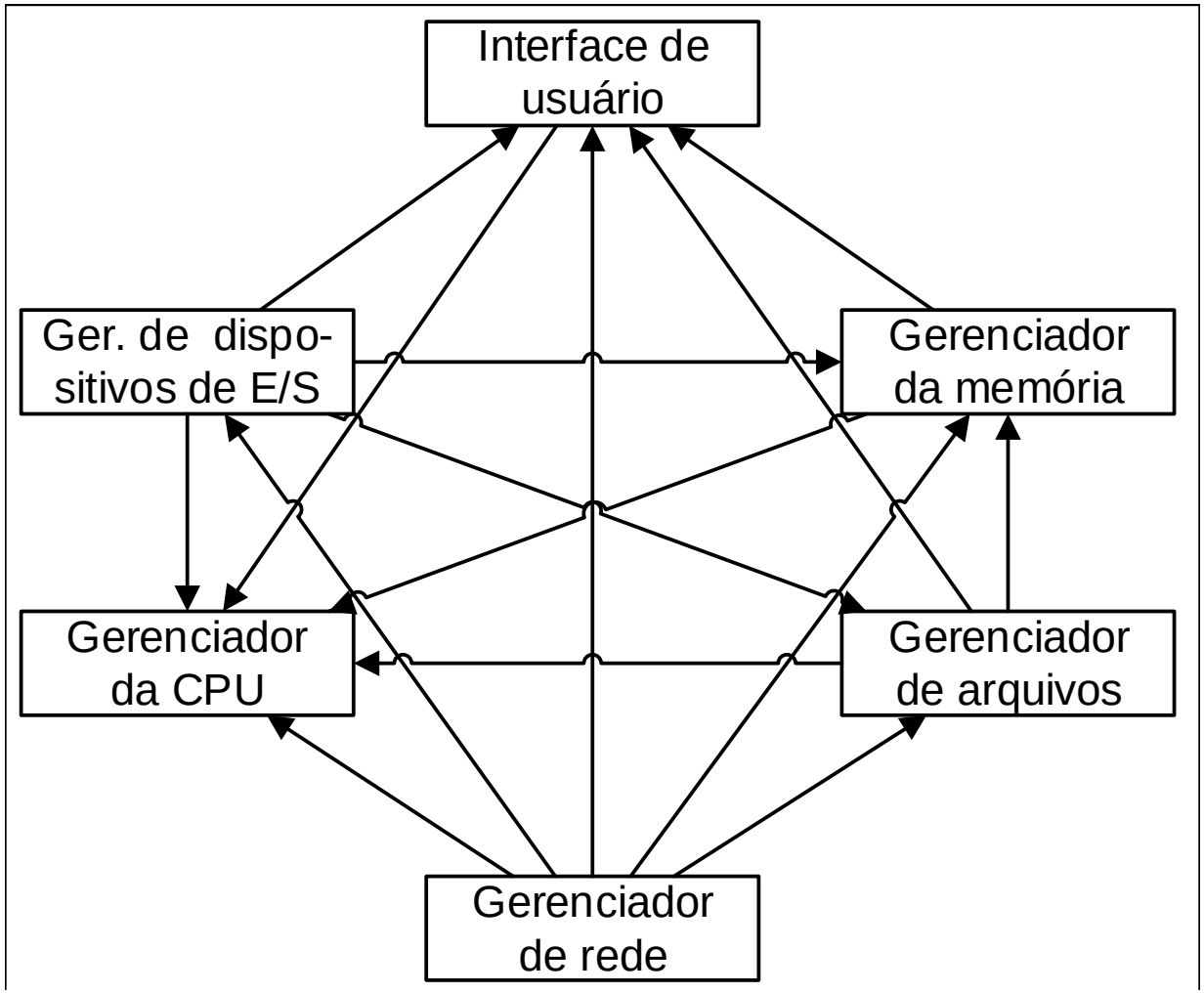


Figura: Componentes de um SO e a relação entre eles

Componentes de um SO - Interface de usuário (*User Interface* – UI ou *shell*)

- Componente do SO responsável por receber e interpretar comandos do usuário e exibir resultados na tela, se for o caso;
- Basicamente há dois tipos de UIs:
  - *Graphical User Interface* – GUI;
  - *Text-based User Interface* – TUI, ou *Console User Interface* – CUI.

- Responsável por *alocar* memória RAM de acordo com as necessidades;
- Evitar que falte memória RAM para os programas;
- Há dois tipos de gerenciadores de memória: **Monoprogramação** e **Multiprogramação**
- Monoprogramação:
  - A maior parte da RAM é alocada para um único processo e a menor parte para manter o SO;
  - Todo o programa está em memória RAM até o fim de sua execução;
  - O programa deve-se ajustar à quantidade de memória RAM;
  - Enquanto o programa está em execução, nenhum outro pode rodar; é um uso muito ineficiente de CPU e RAM.

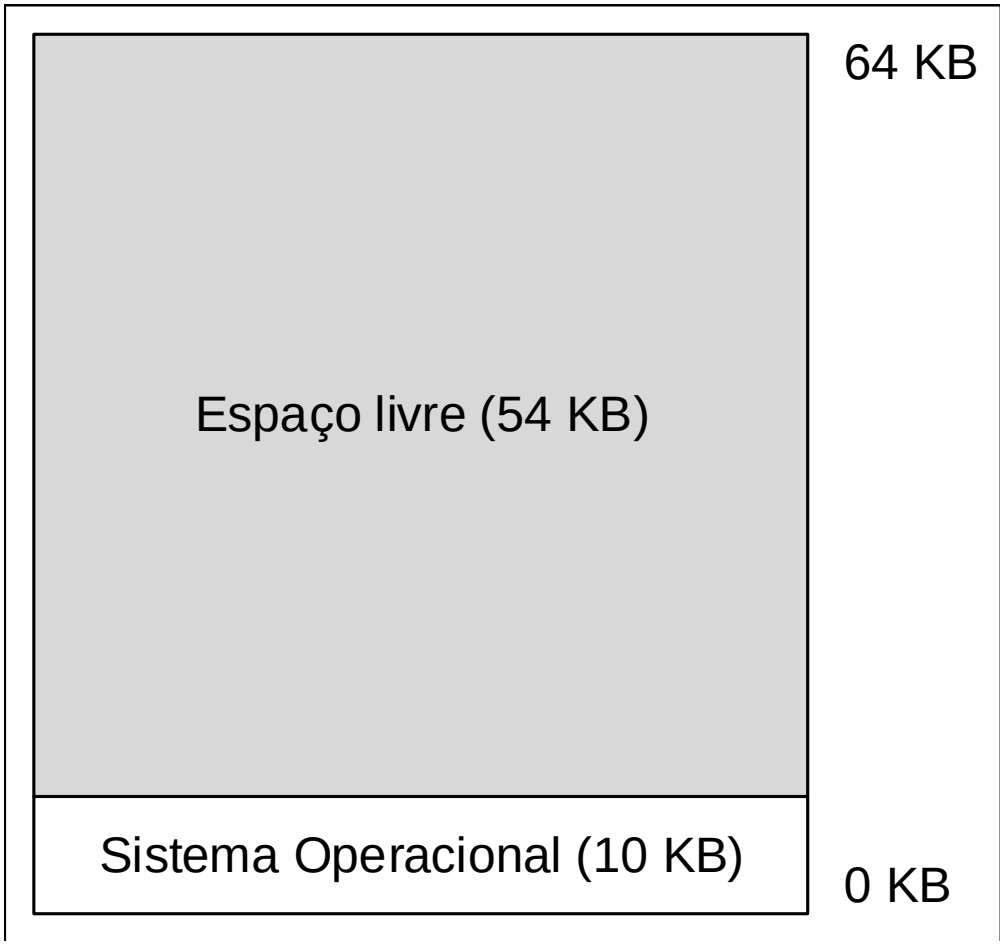


Figura: Monoprogramação - apenas no *shell*

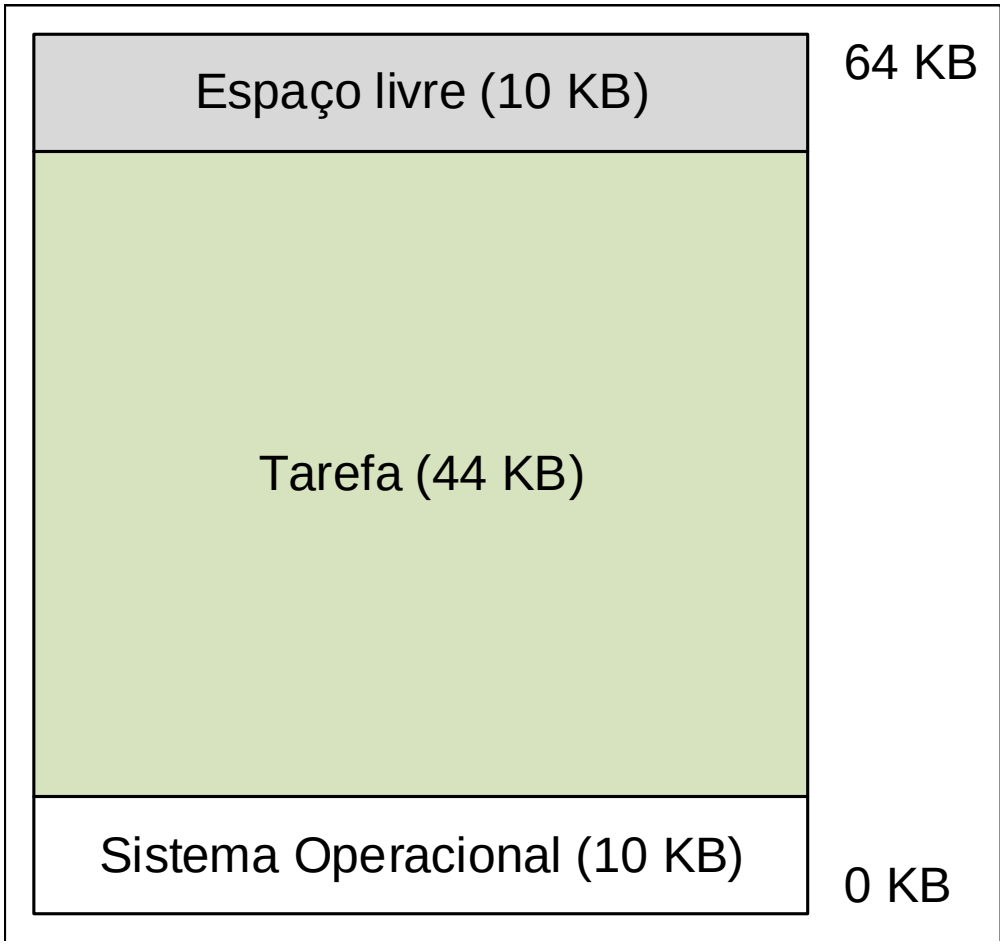


Figura: Monoprogramação - um programa aberto

- Multiprogramação:
  - A memória RAM é alocada para diversos processos;
  - Os processos *competem/concorrem* pelo tempo de CPU;
  - Há dois grandes tipos de multiprogramação: com ou sem *swap* (memória virtual);
  - Sem memória virtual:
    - Partições (ou particionamento);
    - Paginação;
    - Segmentação.
  - Com memória virtual:
    - Paginação sob demanda;
    - Segmentação sob demanda.

- Particionamento:
  - Não utiliza memória virtual – o processo fica todo em memória RAM;
  - A memória RAM é dividida em *partições* de tamanho variável;
  - A CPU executa algumas instruções de um processo até encontrar um comando de E/S ou até o seu fim do tempo de CPU;
  - Antes de mudar para o próximo processo, o SO *guarda o contexto daquele processo*;
  - Pode haver estabelecimento de prioridades
  - Os processos são postos em espaços *contíguos* de memória.

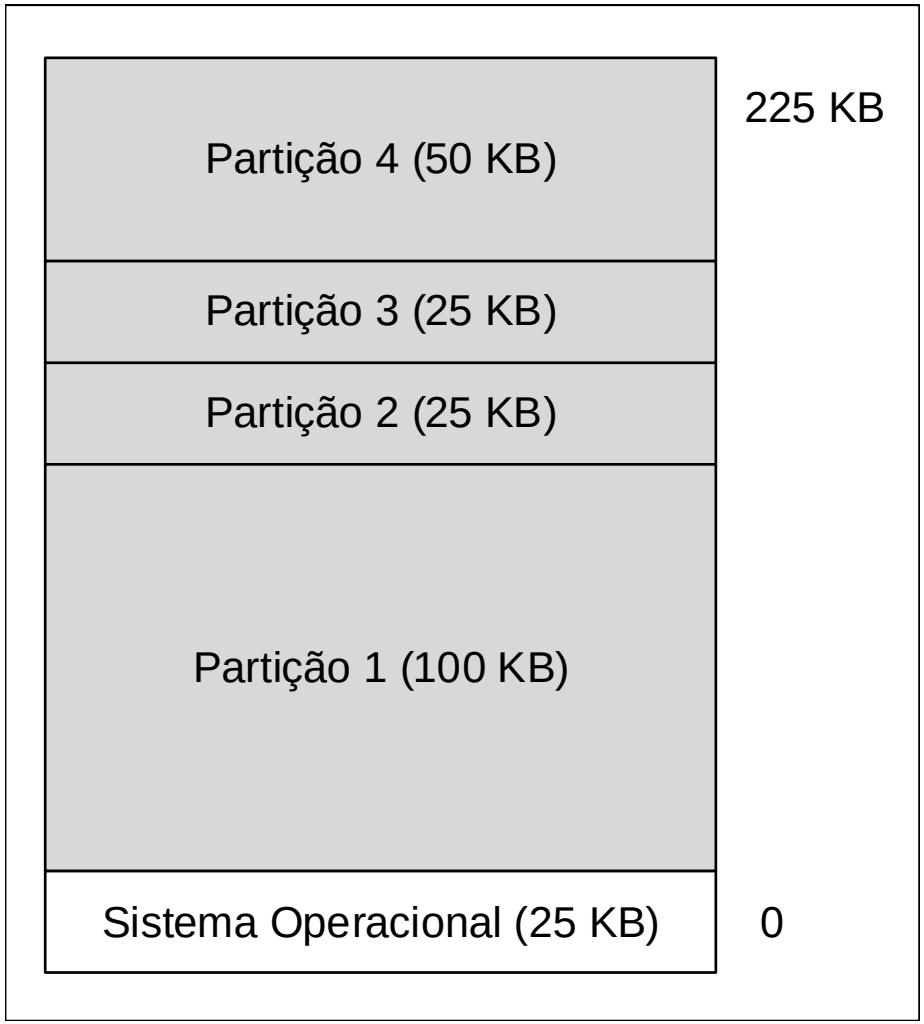


Figura: Multiprogramação/partições fixas - nenhum programa aberto

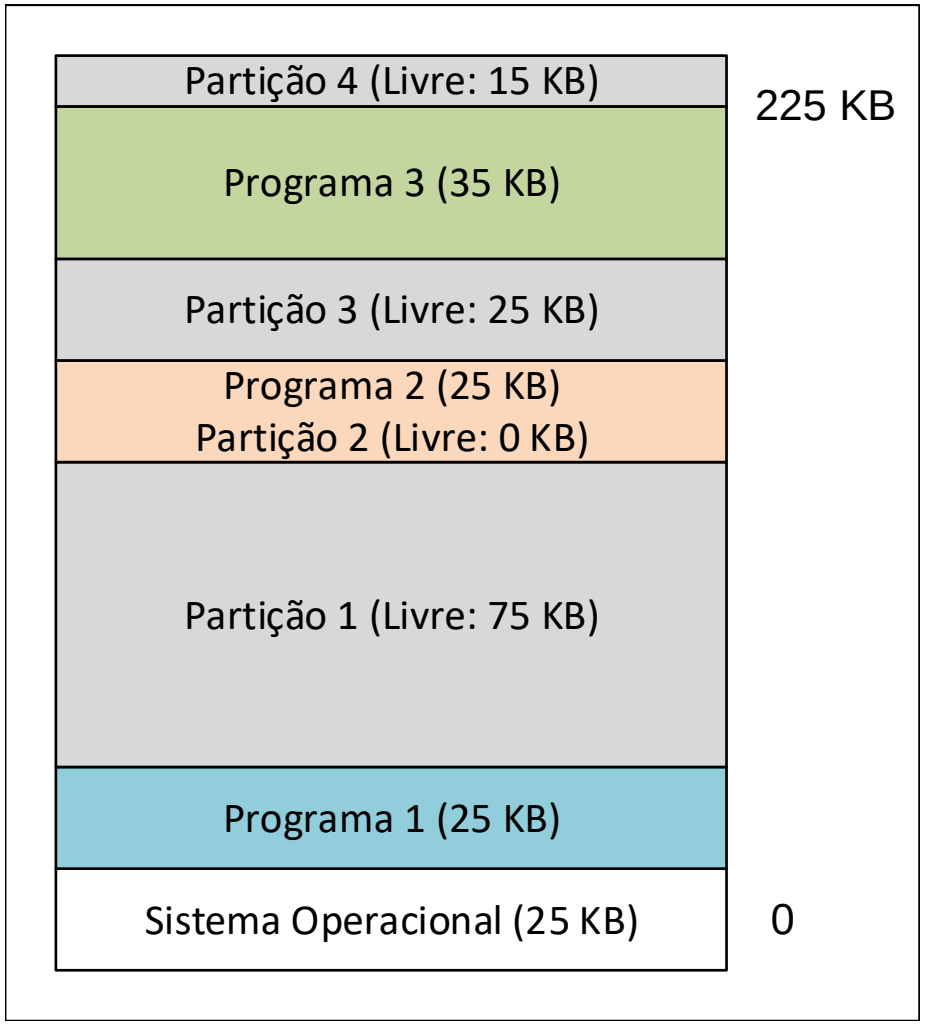


Figura: Multiprogramação/partições fixas - três programas abertos

- Paginação:
  - Não utiliza memória virtual;
  - A memória RAM é dividida em *quadros* de tamanho fixo;
  - Os processos são divididos em *páginas* com o mesmo tamanho dos quadros;
  - Os processos não precisam ser postos em espaços contíguos de memória.

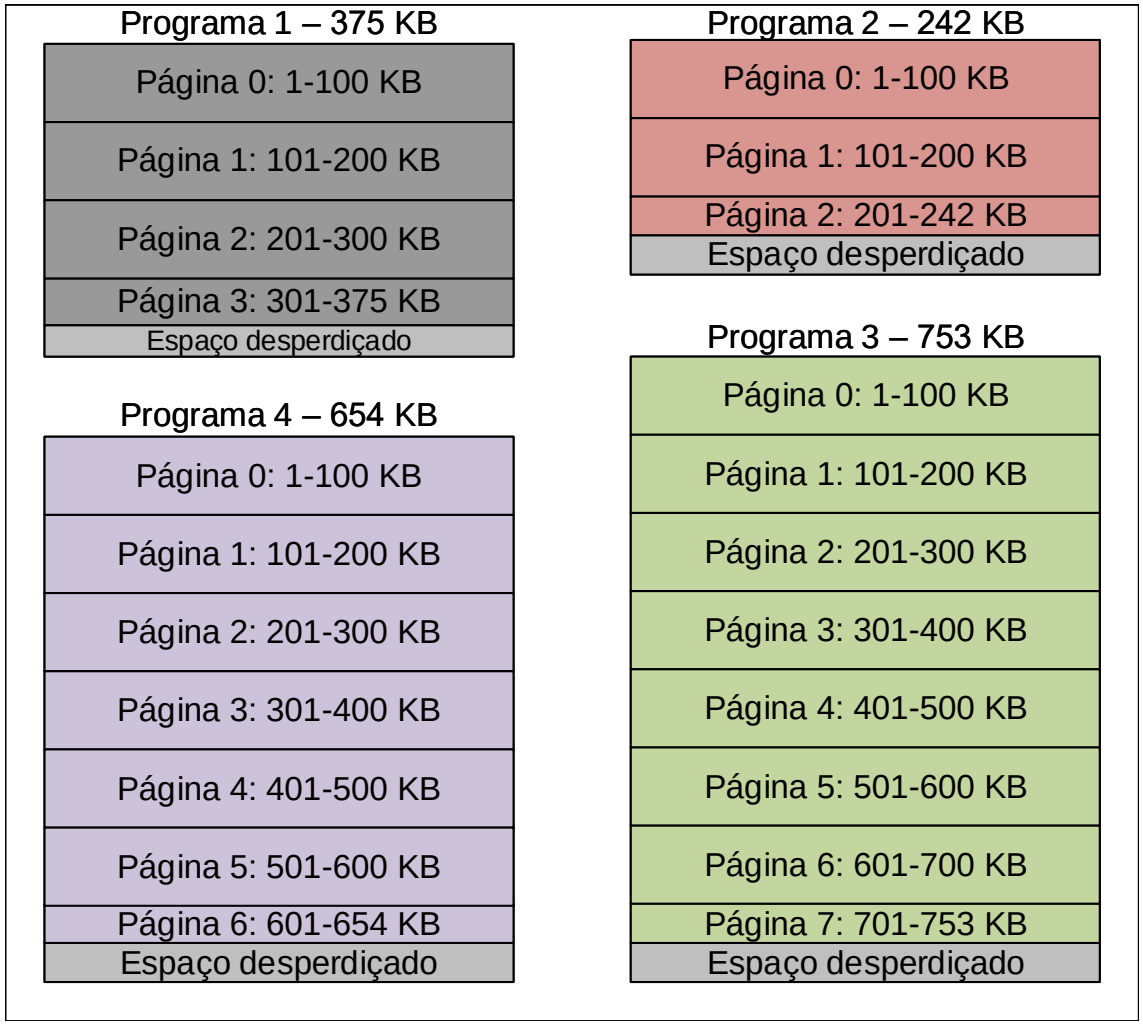


Figura: Multiprogramação/paginação - quatro programas divididos em páginas

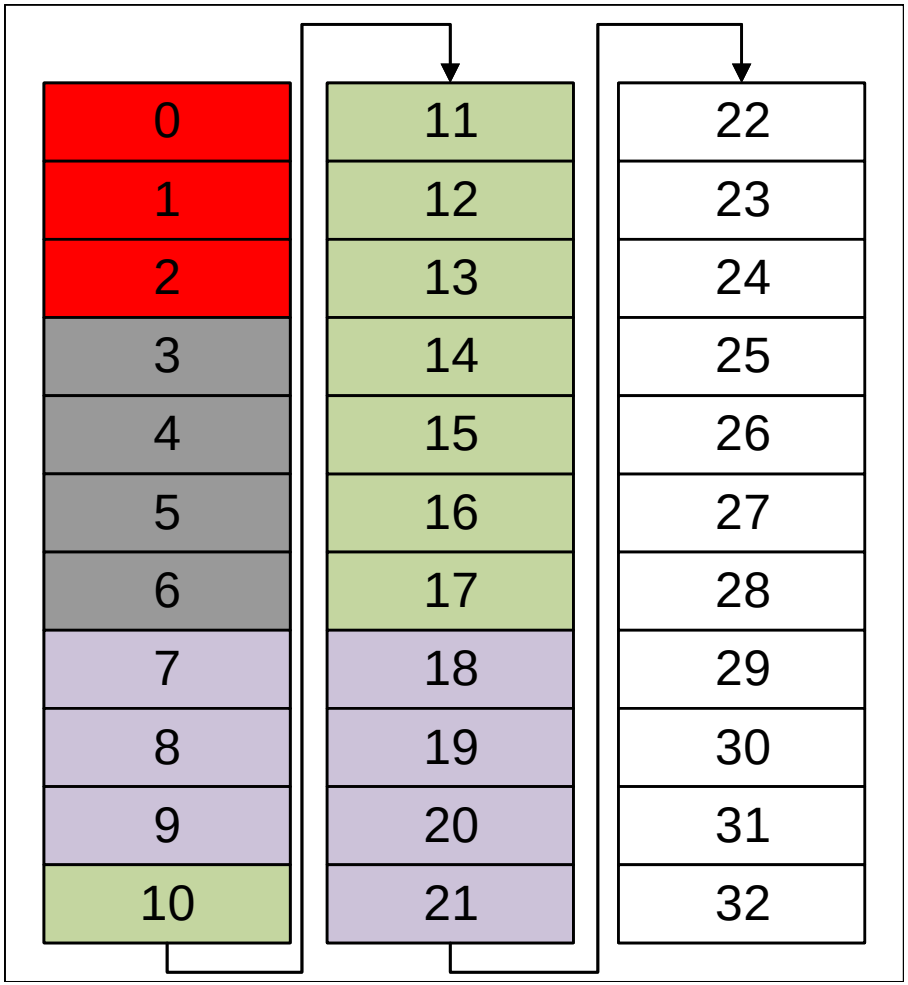
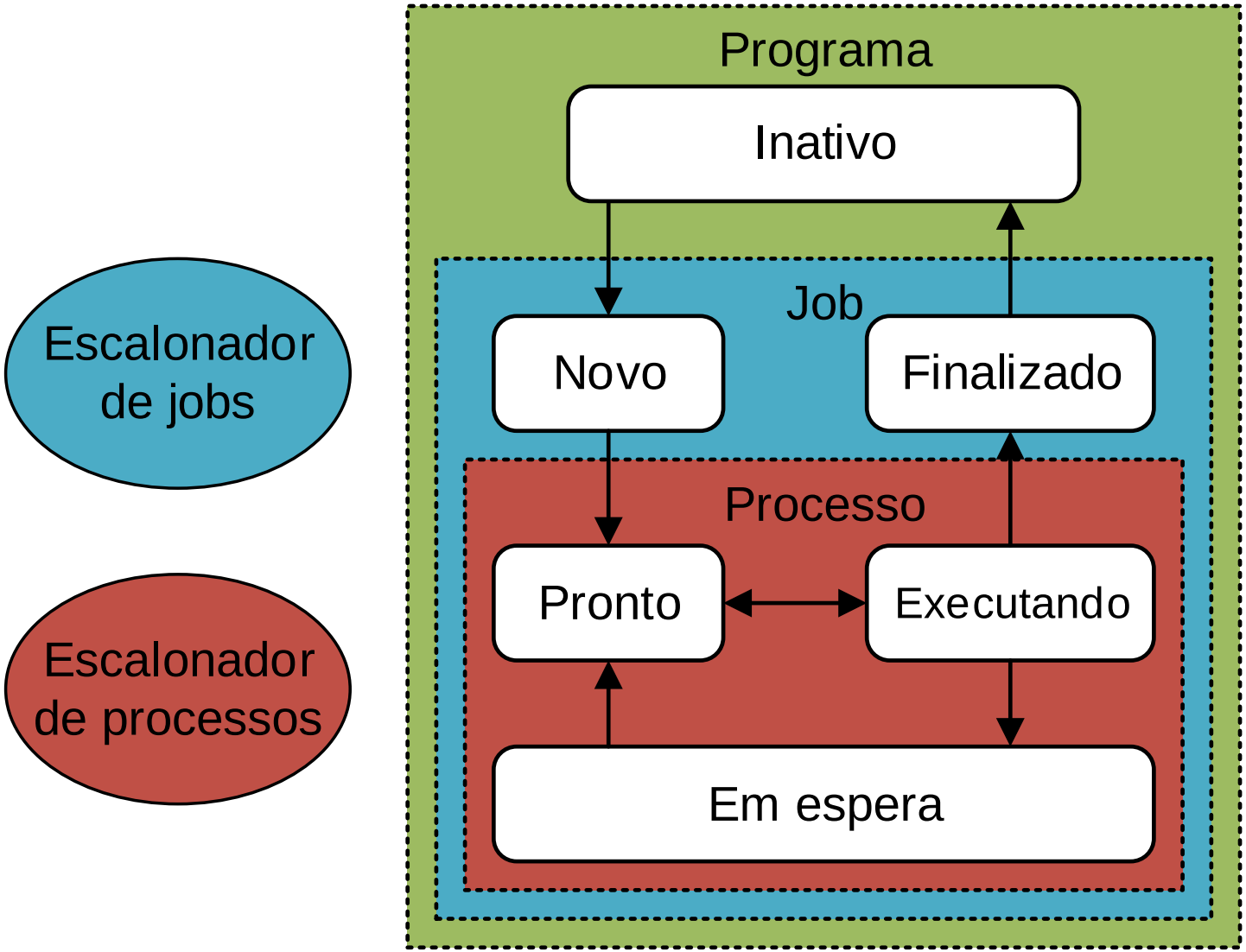


Figura: Multiprogramação/paginação - alocação de memória para três programas e o SO

- Paginação sob demanda:
  - Utiliza memória virtual;
  - Uma diferença básica entre *Paginação* e *Paginação sob demanda* é que apenas as partes *necessárias* para a execução do processo são carregadas na RAM;
  - As partes restantes dos processos que não estão na RAM estão na memória virtual;
  - Um SO pode remover parte dos processos na RAM e devolvê-lo à memória virtual segundo algum algoritmo.

- Programa é um fluxo de instruções armazenado em memória secundária e que não está ativo;
- *Job* é um programa que foi colocado na fila de espera para carregamento na memória primária e/ou para execução;
- Processo é um programa que está em execução pela CPU;

Componentes de um SO - Gerenciador de processos



- É responsável por gerenciar eficientemente o acesso aos dispositivos de E/S;
- O gerenciamento é feito utilizando:
  - O monitoramento de cada dispositivo de E/S; se está funcionando corretamente; se está livre ou ocupado;
  - Uma fila pada cada dispositivo de E/S;
  - O controle de acesso e das políticas de acesso de cada dispositivo de E/S (FIFO, FILO etc).
- É importante ressaltar que o Gerenciador de E/S se comunica com dispositivos através dos *drivers*!

- É responsável por criar, excluir, modificar e nomear arquivos;
- Supervisiona a alocação do espaço livre de forma que outros arquivos não sejam sobrescritos inadvertidamente;
- Controla o acesso aos arquivos; se determinado usuário ou programa pode ler, gravar e executar certos arquivos;

Qual a importância de um SO para um programador?

- Notem que os SOs oferecem serviços para os *softwares* aplicativos – fornecem um modelo **simples** de computador:
  - O Gerenciador de processos lida com as complicações de atribuir processos e **threads** às CPUs;
  - O Gerenciador de memória lida com a quantidade de memória frequentemente menor do que o total exigido pelos processos;
  - O Gerenciador de arquivos oferece uma interface simples para se manipular arquivos;
  - O Gerenciador de E/S oferece interfaces padronizadas para acessar os diversos tipos de dispositivos de E/S.

Qual a importância de um SO para um programador?

- Esses serviços essenciais do SO se situam no *kernel*, que adiciona uma camada de abstração o hardware
- O SO oferece APIs (Interfaces de Programação de Aplicativos, do inglês *Application Programming Interfaces*) para os softwares aplicativos;
- Os programadores não precisam necessariamente saber de detalhes arquiteturais da máquina que se está programando;
- Além do SO, temos os compiladores.

Qual a importância de um SO para um programador?

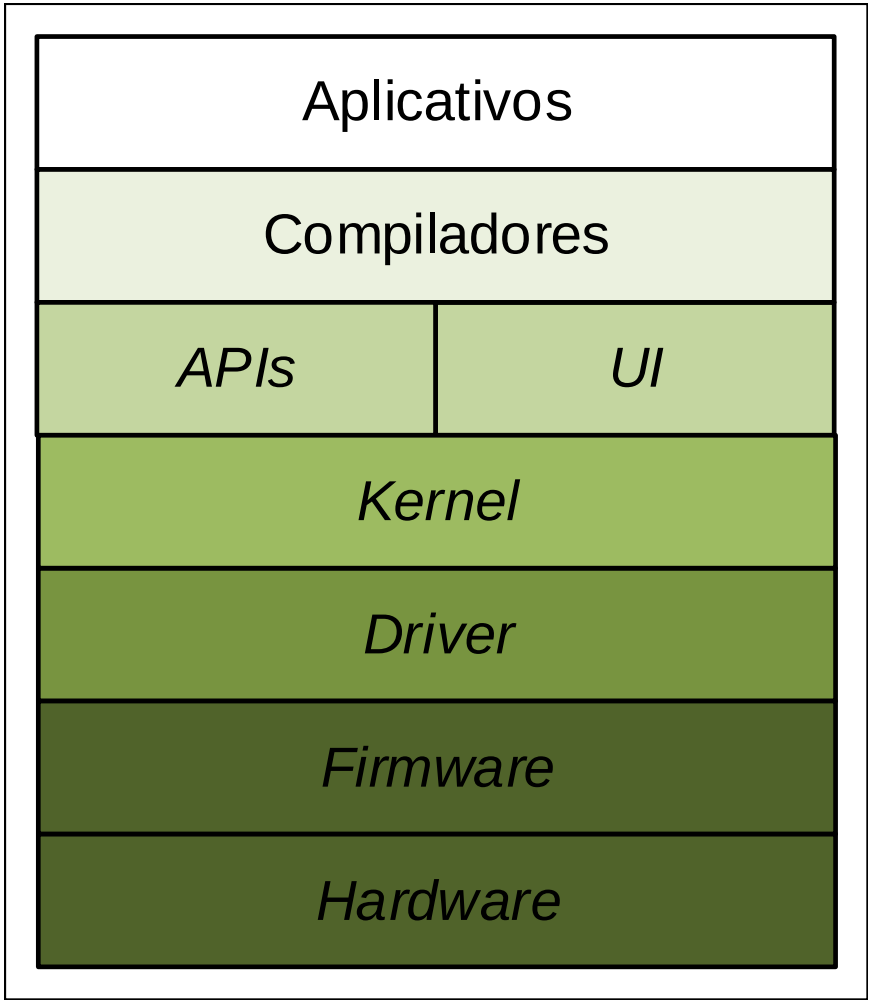


Figura: Visão de um sistema computacional em camadas

Compiladores

- Com arquitetura de John von Neumann, o programador programava o computador armazenando o programa em memória;
- Usava linguagem de máquina, usando 1's e 0's;
- Com o aumento dos programas, aumentava a dificuldade de se programar assim:
- O programador fazia programas específicos para aquele computador – cada computador tinha a sua própria arquitetura e o seu processador, seu conjunto de instruções;
- Foi proposto o uso de símbolos (mnemônicos) para se representar padrões binários – surgiram os primeiros montadores (do inglês *assemblers*).

Compiladores

- Com o advento dos compiladores (e dos sistemas operacionais) os programadores passaram a se afastar do "mundo da máquina";
- Programadores não precisavam mais saber de tantos detalhes arquiteturais da máquina;
- Programação de computadores passou a ser um pouco mais fácil.

Example (Programa em linguagem de máquina em bits)

```
0010 0000 1100 0000 0000 0011
1111 0000 0000 0110
0010 0000 0101 0010 0000 0011
0100 1100 1011 0000 0000 0010
0110 0000
1010 1001 0000 0000
1000 1101 1111 1011 0000 0011
1000 1101 1111 1100 0000 0011
0010 0000 1100 0110 0000 0011
```

Example (Programa em linguagem de máquina em hexadecimal)

```
02B0 20 C0 03
02B3 F0 06
02B5 20 52 03
02B8 4C B0 02
02BB 60
02BC A9 00
02BE 8D FB 03
02C1 8D FC 03
02C4 20 C6 03
```

Example (Programa em linguagem de montagem)

```
JSR read_label_detector
BEQ got_first_label
JSR advance_one_step
JMP finde_first_label
RTS
LOA #0
STA label_length
STA label_length + 1
JSR read_flip_flop
```

Example (Programa em linguagem de C)

```
#include <stdio.h>
function SomaDeDoisValores(){
    int SOMA, A, B;
    printf("Digite um numero: ");
    scanf("%i", &A);
    printf("Digite outro numero: ");
    scanf("%i", &B);
    SOMA = A + B;
    printf("O resultado da soma é: ", SOMA);
}
```

Example (Programa em pseudocódigo)

```
Algoritmo "SomaDeDoisValores";
VARIÁVEL:
    SOMA, A, B: inteiro;
INICIO
    ESCREVA("Digite um numero: ");
    LEIA(A);
    ESCREVA("Digite outro numero: ");
    LEIA(B);
    SOMA = A + B;
    ESCREVA(SOMA);
FIM
```

Site interativo sobre compiladores

`http://gcc.godbolt.org/`